Modellierung und Analyse mobiler Architekturen

Volker Gruhn und Clemens Schäfer
Lehrstuhl für Angewandte Telematik / e-Business,* Institut für Informatik,
Fakultät für Mathematik und Informatik, Universität Leipzig
{gruhn, schaefer}@ebus.informatik.uni-leipzig.de

Abstract: Das Verhalten eines mobilen Systems wird bestimmt durch seine Architektur (statische und dynamische Anteile, Softwareverteilung), die zu Grunde liegende Netzwerkinfrastruktur (Topologie, Parameter wie Bandbreiten oder Latenzzeiten) und Interaktionen der Benutzer mit dem System. Um bereits zur Entwurfszeit zu bestimmen, ob ein mobiles System nichtfunktionale Anforderungen wie Antwortzeiten oder Verfügbarkeit von Diensten erfüllt, kann eine Simulation des Systems auf Basis eines Architekturmodells unter Einbeziehung eines Netzwerk- und eines Benutzerinteraktionsmodells durchgeführt werden. Ein derartiger Ansatz unter Verwendung der Architekturbeschreibungssprache Con Moto wird in diesem Beitrag vorgestellt.

1 Motivation

Die Verwendung einer domänenspezifischen Architekturbeschreibungssprache (Architecture Description Language, ADL) zur Modellierung der Architektur mobiler verteilter Systeme wird als sinnvoll angesehen [GS05], denn der Einfluss der Mobilität betont die Notwendigkeit, funktionale wie auch nichtfunktionale Eigenschaften einer Softwarearchitektur zu beschreiben und zu untersuchen. Dies korrespondiert mit der Tatsache, dass Mobilität als das totale Abschmelzen aller Stabilitätsannahmen, die beim Distributed Computing üblicherweise getroffen werden, aufgefasst werden muss [RPM00] und deutet damit auf die Probleme hin, die bei der Entwicklung mobiler Systeme zu lösen sind. Zu den typischen Problemen zählen hierbei Netzwerkstrukturen, die nicht länger statisch sind und bei denen Kommunikationsknoten entstehen und verschwinden können, Kommunikationsfehler auf Grund von Verbindungsabbrüchen in Drahtlosnetzwerken oder eingeschränkte Konnektivität wegen (zu) niedriger Bandbreiten mobiler Kommunikationskanäle. Alle diese Probleme haben gemeinsam, dass sie die zur Laufzeit entstehenden nichtfunktionalen Eigenschaften des mobilen Systems wie Performanz, Robustheit oder Dienstgüte nachhaltig beeinflussen.

Mit unserer ADL *Con Moto* (italienisch für “mit Bewegung”) schlagen wir eine Sprache vor, die es Systementwicklern erlaubt, die genannten Aspekte bereits zur Entwurfszeit zu adressieren und adäquate Entwurfsentscheidungen für das mobile System zu treffen. Die Herausforderung bei der Modellierung mobiler Systeme liegt in der Wahl eines angemes-

*Der Lehrstuhl für Angewandte Telematik / e-Business ist ein Stiftungslehrstuhl der Deutschen Telekom AG.

senen Abstraktionsniveaus, denn Übersimplifizierung würde zu bedeutungslosen Simulationsergebnissen führen, zu komplexe Modelle wären hingegen zur Entwurfszeit unpraktikabel. Zudem sollte der Modellierungsansatz so abstrakt und unabhängig wie möglich von konkreten technischen Realisierungen mobiler Systeme sein, jedoch dürfen technologische Implikationen mobiler Systeme nicht völlig außer Acht gelassen werden, wenn aussagekräftige Simulationsergebnisse erzielt werden sollen.

2 Verwandte Arbeiten

ADLs stellen ein gut erforschtes Thema dar. Die Modellierung nichtfunktionaler Eigenschaften wurde bereits bei Shaw und Garlan [SG95] als relevant angesehen. Die Klassifikationsarbeiten von Medvidovic und Taylor [MT00] bieten einen breiten Überblick über die Eigenschaften verschiedener ADLs, wobei deutlich wird, dass die dort betrachteten ADLs keine Unterstützung für mobile Systeme bieten. Aus diesem Grund sind in letzter Zeit mobile ADLs entstanden [Oqu04, ITLS04], die die Dynamik mobiler Systeme abbilden können, was aus ihrem Ursprung im π-Kalkül herrührt; beide bieten jedoch keine Unterstützung nichtfunktionaler Eigenschaften. Neben den mobilen ADLs existieren weitere Arbeiten im Bereich nichtfunktionaler Eigenschaften. Sie basieren meist auf Lamports TLA+ [Lam02], einer Logik zur Spezifikation nebenläufiger reaktiver Systeme. Zschaler [Zsc04] stellt eine Spezifikation temporaler Eigenschaften von komponentenbasierten Systemen vor, aber diese – ebenso wie die zu Grunde liegenden Arbeiten von Aagedal [Aag01] – lassen die Unterstützung von Mobilität vermissen. Andere Ansätze auf Basis von Markoffketten oder Prozessalgebren (z.B. die Arbeiten von Hermanns und Katoen [HK01]) greifen in diesem Aspekt ebenfalls zu kurz.

3 Con Moto im Überblick

Abbildung 1 zeigt die verschiedenen Elemente von Con Moto. Die eigentliche Architekturbeschreibung (Architectural Description) besteht aus einer Spezifikation der Struktur (Structural Specification) und der Dynamik (Behavioral Specification) des Systems. Zusammen mit Instantiierungsinformationen (Instantiation Information) kann der Simulator das Architekturmodell instantiieren und auf Basis eines Modells des Kommunikationsnetzwerks (Network Model) unter Berücksichtigung von Benutzerinteraktionsmustern (Usage Pattern) die gewünschten Eigenschaften der Architektur bestimmen.

Verhaltensmodell Wie andere ADLs stützen wir das Verhaltensmodell auf Milners π-Kalkül [Mil99] ab. Beim π-Kalkül handelt es sich um eine Prozessalgebra mit expliziter Mobilitätsunterstützung, bei der Prozesse über so genannte Namen kommunizieren. Analog zur Programmiersprache Pict [PT00], die auf dem π-Kalkül basiert, verwenden wir nur eine Untermenge des vollen π-Kalküls, was die (unbeabsichtigte) Modellierung von Nichtdeterminismen verhindert. Bei den Nachrichten, die zwischen den Prozessen

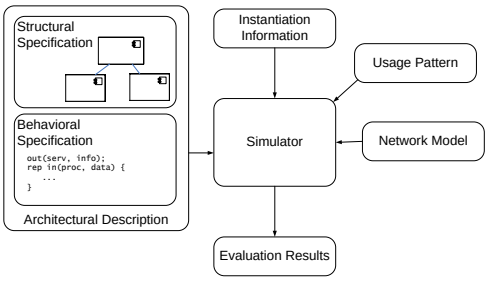

Abbildung 1: Bestandteile von Con Moto

kommuniziert werden, erlauben wir die Annotation einer simulierten Paketgröße, um die Transportzeiten von Nachrichtenpaketen über das Netzwerk abbilden zu können.

Strukturelles Modell Wie bei ADLs allgemein üblich, besteht das strukturelle Modell aus Komponenten, Konnektoren und Konfigurationen.

Die Komponenten stellen den Ort dar, an dem die eigentlichen Berechnungen stattfinden. Bei Con Moto unterscheiden wir explizit zwischen *physikalischen* und *logischen* Komponenten. Physikalische Komponenten bilden Geräte ab, also Hardware mit Ressourcenbeschränkungen, während logische Komponenten die eigentlichen Softwarekomponenten repräsentieren, die sowohl als Komponenten (zustandslos) als auch als Komponenteninstanzen (zustandsbehaftet) vorkommen können.

Konnektoren bilden die Interaktionen zwischen Komponenten ab. In Con Moto unterscheiden wir dabei explizit zwischen *physikalischen* und *logischen* Konnektoren. Logische Konnektoren sind ideal und stellen die Kommunikationsbeziehungen zwischen logischen Komponenten dar, wogegen physikalische Konnektoren über eine beschränkte Bandbreite und Latenzzeiten größer Null verfügen und die Kommunikation zwischen physikalischen Komponenten realisieren. Logische Konnektoren sind immer in physikalische Konnektoren eingebettet.

Durch diese Art der Modellierung und die Möglichkeit, logische Komponenten wie auch logische Konnektoren zwischen Prozessen zu kommunizieren, können alle in [MPR99] genannten Arten von Mobilität (Client-server, Remote evaluation, Code-on-demand und Mobile agents) realisiert werden.

4 Anwendungsbeispiel

Im Folgenden präsentieren wir ein einfaches mobiles Client/Server-System. Die Benutzer des Systems verfügen über mobile Endgeräte, die über jeweils einen Mobilfunkkanal mit einem Server verbunden sind; hierfür sehen wir entweder eine GPRS-Verbindung mit relativ niedriger Bandbreite oder eine UMTS-Verbindung mit entsprechend höherer Bandbreite vor. In unserem Beispielsystem existieren drei Softwarekomponenten: die Benutzero-

berfläche (UI) wird auf den mobilen Endgeräten verteilt. Die Persistenzkomponente (DATA) ist nur auf dem Server verfügbar, wogegen die Komponente mit der Geschäftslogik (BUSINESS) entweder auf dem Server oder aber auf den mobilen Endgeräten verteilt sein kann. Ruft der Benutzer über die Benutzeroberfläche einen Dienst auf, sendet die Komponente UI eine Dienstanfrage an die Komponente BUSINESS, die ihrerseits wiederum einen Service von DATA aufruft. Wenn dieser ein Ergebnis zurückliefert, gibt BUSINESS dieses an UI weiter, wo das Ergebnis dem Benutzer angezeigt wird. Die Struktur dieses Systems wird im linken Teil der Abbildung 2 verdeutlicht.

4.1 Modellierung in Con Moto

In der Con Moto-Beschreibung dieses Systems (einem XML-Dokument) werden zwei physikalische Komponenten `MOBILE` und `SERVER` deklariert, die jeweils mit Rechenleistung und möglichen Netzwerkverbindungen, die während der Simulation zu physikalischen Konnektoren führen, parametrisiert werden. `MOBILE` Komponenten können sich daher entweder mit Netzwerkknoten vom Typ UMTS oder GPRS verbinden, während für die `SERVER` Komponente nur eine Verbindung mit einem WAN möglich ist.

Im Netzwerkmodell werden die Netzwerktypen UMTS, GPRS und WAN definiert und die Bandbreiten (10.0, 2.0 und 1000.0 kBit/s) gesetzt. Ein zusätzlicher Netzwerkknoten namens `backbone` dient zur Verknüpfung aller Kommunikationsinstanzen und erlaubt so die direkte Adressierung aller physikalischer Komponenten, wie dies im Internet durch IP-Adressen auch möglich ist.

Durch die Einführung von Ports und Porthierarchien werden Schnittstellen zum Veröffentlichen und Aufrufen von Methoden definiert. Entsprechende Makros in diesen Schnittstellen realisieren die Kommunikationsprotokolle für die Dienstaufrufe im π-Kalkül. Durch diese Makros und deren Wiederverwendung in Verbindung mit den Porthierarchien ist es möglich, in Con Moto das eigentliche System strukturell äquivalent zu einer imperativen Programmbeschreibung zu definieren.

Für die logischen Komponenten `DATA`, `BUSINESS` und `UI`, die ebenfalls Bestandteil des Con Moto-Modells sind, werden Start-Up Prozesse definiert, die bei Instantiierung der Komponenten ausgeführt werden und die entsprechenden Lookup- und Konnektivitätsfunktionen durchführen. Die eigentlichen Services auf den logischen Komponenten werden ebenfalls in Prozessen im π-Kalkül ausgedrückt und sind im Beispiel hier durch das Blockieren der CPU für eine bestimmte Zeit (100 ms bei `DATA` und 500 ms bei `BUSINESS`) und das Zurückliefern von Datenblöcken definierter Größe (100 Byte bei `DATA` und 5000 Byte bei `BUSINESS`) modelliert.

4.2 Simulation

Das hier beschriebene Beispielsystem wurde mit Hilfe unseres Con Moto-Simulators einer Simulation mit 10 bis 150 Benutzern, also mit 10 bis 150 MOBILE Geräten, unterzogen.

Die Benutzerinteraktionen mit dem System wurden durch einen Poissonprozess mit einer Ankunftsrate von 10 Ereignissen pro Stunde modelliert. Die gesamte Simulation wurde mit einer Zeitauflösung von einer Millisekunde durchgeführt. Abbildung 2 zeigt das Simulationsergebnis, wobei die wesentliche Aussage ist, dass ab 90 Benutzern das System zunehmend überlastet ist und die Antwortzeiten der Dienste erheblich ansteigen. Die Antwortzeiten für GPRS und UMTS verhalten sich hierbei leicht unterschiedlich, was auf die höhere Bandbreite bei UMTS im Vergleich zu GPRS zurückzuführen ist.

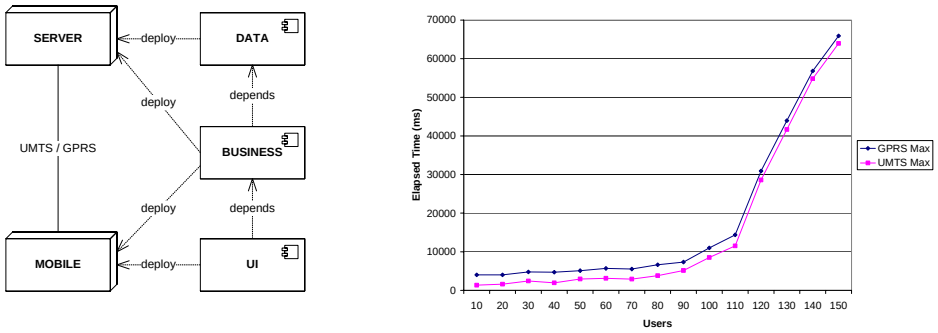

Abbildung 2: Beispielsystem und Simulationsergebnisse

5 Diskussion und Ausblick

In diesem Beitrag haben wir beschrieben, wie mobile Systeme mit Con Moto mit dem Ziel modelliert werden können, zur Entwurfszeit Dienstgüteparameter des Systems durch Simulation zu bestimmen. Indem wir eine Architekturbeschreibung auf Basis des π-Kalküls vorgenommen und dabei eine klare Unterscheidung zwischen physikalischen und logischen Komponenten und Konnektoren vorgenommen haben, ist die Modellierung der mobilen Systeme auf einem recht hohen Abstraktionsniveau mit vertretbarem Aufwand möglich. Erste Simulationsversuche für ein einfaches Beispiel zeigen, dass der Ansatz allgemein Erfolg versprechend ist.

Weitergehende Arbeiten umfassen die Untersuchung von Modellen für physikalische Kommunikationskanäle, die bislang lediglich durch die Annahme einer konstanten Bandbreite und Latenzzeit modelliert werden. Hierbei sind komplexere Modelle des Übertragungsverhaltens denkbar und für realistische Simulationsergebnisse auch nötig, die schwankende Bandbreiten, unterschiedliche Latenzzeiten oder sogar Verbindungsabbrüche zulassen. Ebenso im Bereich der Modellierung von Benutzerinteraktionen mit dem System ergibt sich weiterer Forschungsbedarf, wobei jedoch nicht nur stochastische Prozesse tiefere Betrachtung erfahren sollten, sondern auch Aspekte, wie die Interaktionsmuster für die Simulation aus gegebenenfalls vorhandenen Geschäftsprozessmodellen abgeleitet werden können. Eine weitere zukünftige Aufgabe ist schließlich die Evaluierung des vorgestellten Ansatzes durch Vergleich von Simulationsergebnissen mit belastbaren Messergebnissen aus tatsächlich implementierten Systemen.

Literatur

[Aag01] Jan Øyvind Aagedal. *Quality of Service Support in Development of Distributed Systems*. Dissertation, University of Oslo, 2001.

[GS05] Volker Gruhn und Clemens Schäfer. Architecture Description for Mobile Distributed Systems. In *Proceedings of the Second European Workshop on Software Architecture (EWSA 2005)*, Seiten 239–246. Springer-Verlag Berlin Heidelberg, 2005.

[HK01] Holger Hermanns und Joost-Pieter Katoen. Performance Evaluation := (Process Algebra + Model Checking) × Markov Chains. In *Proceedings of CONCUR 2001*, LNCS 2154, Seiten 59–81. Springer-Verlag Berlin Heidelberg, 2001.

[ITLS04] Valérie Issarny, Ferda Tartanoglu, Jinshan Liu und Françoise Sailhan. Software Architecture for Mobile Distributed Computing. In *Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA'04)*, Seiten 201–210. IEEE, 2004.

[Lam02] Leslie Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, 2002.

[Mil99] Robin Milner. *Communicating and Mobile Systems: the π-Calculus*. Cambridge University Press, 1999.

[MPR99] Cecilia Mascolo, Gian Pietro Picco und Gruia-Catalin Roman. A fine-grained model for code mobility. In *ESEC/FSE-7: Proceedings of the 7th European software engineering conference held jointly with the 7th ACM SIGSOFT international symposium on Foundations of software engineering*, Seiten 39–56, London, UK, 1999. Springer-Verlag.

[MT00] Nenad Medvidovic und Richard N. Taylor. A Classification and Comparison Framework for Software Architecture Description Languages. *IEEE Transactions on Software Engineering*, 26(1):70–93, Januar 2000.

[Oqu04] Flavio Oquendo. π-ADL: An Architecture Description Language based on the Higher-Order Typed π-Calculus for Specifying Dynamic and Mobile Software Architectures. *ACM Software Engineering Notes*, 29, Mai 2004.

[PT00] Benjamin C. Pierce und David N. Turner. Pict: A Programming Language Based on the Pi-Calculus. In G. Plotkin, C. Stirling und M. Tofte, Hrsg., *Proof, Language and Interaction: Essays in Honour of Robin Milner*. MIT Press, 2000.

[RPM00] Gruia-Catalin Roman, Gian Pietro Picco und Amy L. Murphy. Software Engineering for Mobility: A Roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, Seiten 241–258. ACM Press, 2000.

[SG95] Mary Shaw und David Garlan. Formulations and Formalisms in Software Architecture. In Jan van Leeuwen, Hrsg., *Computer Science Today: Recent Trends and Developments*, Jgg. 1000 of *Lecture Notes in Computer Science*, Seiten 307–323. Springer, 1995.

[Zsc04] Steffen Zschaler. Formal Specification of Non-functional Properties of Component-Based Software. In Jean-Michel Bruel, Geri Georg, Heinrich Hussmann, Ileana Ober, Christoph Pohl, Jon Whittle und Steffen Zschaler, Hrsg., *Workshop on Models for Non-functional Aspects of Component-Based Software (NfC'04) at UML conference 2004*, September 2004.